

Appendix A: Example of Server Executable Code In Recognition/Annotation Process

5

```
bl_annotate.pl
#!/usr/bin/perl -w
10 ######
# Script to annotate the buylites
# Written by Henri
#
#####
15
use strict;
use LWP::UserAgent;
use Apache::Registry;
use Time::HiRes qw (time);
20 ######
# Pull in words from page and compare to hash tree #####
my $url;
my $url_unescape;
my $textdata;
my @tempmatchingids=();      # temp array to return longest matching
word
my %finallist=();           # hash that holds the final matching proper
nouns as its keys
30
my $i=0;
my $j=0;
my $res='';
my @words;
my $max_annotations = 15; # steve - 5.24.01
35
$url = $ENV{'QUERY_STRING'};
$url =~ s/(^|&)ref=([^\&]+)\.*$/$2/o;
$url_unescape = $url;
$url_unescape =~ s/%([0-9A-Fa-f]{2})/chr(hex($1))/eg; # Unescape the
stuff
40 ######
# Get the Page #####
# Need to use LWP::UserAgent instead of LWP::Simple because we need the
timeout function
my $ua = new LWP::UserAgent;
45
$ua->timeout(10); # 10 second timeout
my $request = new HTTP::Request('GET', $url_unescape);
$textdata = $ua->request($request);
if ($textdata->is_success) {
    $textdata = $textdata->content;
```

```
    } else {
        $textdata = '';
    }
##### Done w/ the Page #####
5 my $aa = time;

# $textdata =~ s/<IMG[^>]+>//igo;
$textdata =~ s/<[A|a] .+?\/[A|a]>//go;
$textdata =~ s/<SCRIPT.+?<\//SCRIPT>//igo;
10 $textdata =~ s/<HEAD( |>).+?\//HEAD>//igo;
# $textdata =~ s/<FORM.+?<\//FORM>//igo;
# $textdata =~ s/<object.+?<\//object>//igo;
# $textdata =~ s/<embed.+?<\//embed>//igo;
$textdata =~ s/<[^>]+>//go;
15 $textdata =~ s/&[^;]+;//go;
# $textdata =~ s/\[[^]]+\]//igo;

@words = split /[^a-zA-Z0-9\-\']+/, $textdata;

20 my $wordcount = scalar (@words);
my $numlist = 0;
my $stringtosearch = '';
my $nounid;
my $maxwords = 10;      # maximum number of words in one buylite string
25 ###### BEGIN RECOGNITION #####
while (($i < $wordcount) && ($numlist < $max_annotations)) {

    # Throw out the 1st word if it's not in the first word list
    # This is only to accelerate dramatically the searching
    30 if (not exists ($ANNOTATOR::firstword{$words[$i]})) {
        $i++;
        next;
    }

    35 $j=$i;
    $stringtosearch = '';

    while (($j-$i <= $maxwords) && ($j < $wordcount)) {
40    # Are we not yet at end of file?
        $stringtosearch .= $words[$j];
        if (exists($ANNOTATOR::pnlist{$stringtosearch})) {
            push @tempmatchingids, $stringtosearch;
            $i=$j+1;
        }
        45 $j++;
        $stringtosearch .= ' ';
    }

    50 if (scalar(@tempmatchingids) > 0) {          # did we have
earlier full keyphrase matching?
        $finalist{pop @tempmatchingids} = 1;
    }
}
```

```
        $numlist++;
        @tempmatchingids = ();
    } else {
        $i++;
5     }
} # while
##### END RECOGNITION #####
$aa = time - $aa;
10 print STDERR ("$url_unescape --- $aa\n");

##### Output #####
my $linkstag;
my $cache_header;
15 my $outfile;

if ($numlist == 0) {          # No buylites found
    $outfile = 'function makeChanges() {}; function blHighlight()
{}';
20     goto PRINTOUTPUT;
}

$ENV{'QUERY_STRING'} =~ /(^\&)pn=([^\&])/o;
if ((defined $2) && ($2 eq 'n')) {           # buylites disabled
25     through cookie
        $outfile = 'function makeChanges() {}; function blHighlight() {
blInitUI(BL_HIDE); }';
        goto PRINTOUTPUT;
}
30 # From here onwards we found buylites and they are enabled on the
client

$outfile = << "EOT";

35 function makeChanges() { var i9i; var refurl = '$url'; [LINKS_TAG]
} function blHighlight(){TAG_NAME=new
Array();TAG_NAME[0]='P';TAG_NAME[1]='LI';TAG_NAME[2]='UL';TAG_NAME[3]='
TD';TAG_NAME[4]='DL';var pnshown=0;for(var
i=0;i<TAG_NAME.length;i++){var
40 ps=document.all.tags(TAG_NAME[i]);for(j=0;j<ps.length;j++){g_html=ps(j)
.innerHTML;g_outer=ps(j).outerHTML;g_tag=g_outer.substring(0,g_outer.in
dexOf('>')+1);if((g_html.length>=5000)||((g_html=='')))continue;for(var
p=1;p<pnpatbeg.length;p++)g_html=replacepat(g_html,p);chgflag=0;pntmpli
st='';makeChanges();for(var p=pnpatbeg.length-1;p>0;p--
45 )g_html=expandpat(g_html,p);if(chgflag==1){var
pnrv=1;if(i!=3)pnrv=setOuterHTML(ps(j),g_tag+g_html+'</' +TAG_NAME[i]+'
');else pnrv=setInnerHTML(ps(j),g_html);if(pnrv==0)pnshown=1;}for(var
p=1;p<pnpatbeg.length;p++)pnpattotal[p]=0;}if(pnshown==1)blInitUI(BL_S
HOW);
50 EOT

foreach $res (keys %finalist) {
```

```
#      $linkstag = "changeHTML(\"".$ANNOTATOR::pnlist{$res}."\",
\"http://colostage.bizrate.com/buylites/landing.xpml?keyword=".$ANNOTAT
OR::pnlist{$res}."&aff_id=-1&noun_id=".$res."&rf=buy".$res."|-
1&ref_url=\"$ + refurl);";
5      $linkstag = "changeHTML(\"".$res."\",
\"http://colostage.bizrate.com/buylites/landing.xpml?keyword=".$res."&a
ff_id=-
1&noun_id=".$ANNOTATOR::pnlist{$res}."&rf=buy".$ANNOTATOR::pnlist{$res}
."|-1&ref_url=\"$ + refurl);";
10     $outfile =~ s/\[LINKS_TAG]/$linkstag [LINKS_TAG]/;
}
$outfile =~ s/\[LINKS_TAG]//o;
20
```

PRINTOUTPUT:

```
15 print "HTTP/1.0 200 OK\n";
print "Content-Length: " . length($outfile) . "\n";
print "Cache-Control: max-age=86400\n";
print "Content-type: text/html\n\n";
print $outfile;
```

© BizRate.com

Appendix B: Example of Client Executable Code In Recognition/ Annotation Process

5 The following four server-side components are sent by one or more servers to a client for execution in the following order on the client:

buylites.js (JScript file) [Normally found on content document initially executed on a client system]

bl_style.css (stylesheet)

bl_vb.js (VBScript file)

10 bl_start.js (JScript file)

buylites.js

```
var BL_BURL = 'http://xxx.xxxxx.com/buylites';
var BL_SURL = BL_BURL+'/bl';
var BL_IURL = BL_SURL+'/images';
15 var blNavAgt = navigator.userAgent.toLowerCase();
var blIsMajor = parseInt(navigator.appVersion);
var blIsIE = (blNavAgt.indexOf("msie") != -1);
var blIsIE4 = (blNavAgt.indexOf("msie 4") != -1);
var blIsIE4up = (blIsIE && (blIsMajor >= 4));
20 var blIsWin = (blNavAgt.indexOf("windows ")!=1);

if (blIsIE4up && blIsWin)
{
    if (typeof(BL_AFF_ID) == 'undefined') BL_AFF_ID = -1;
25    document.write('
        <link href="'+BL_SURL+'/bl_style.css" rel="stylesheet type="text/css">\\
        <script language=jscript src="'+BL_SURL+'/bl_main.js"></script>\\
        <script language=VBScript src="'+BL_SURL+'/bl_vb.js"></script>\\
        <script language=jscript
30    src="'+BL_BURL+'/r1.pl?aff_id='+BL_AFF_ID+'"></script>\\
        <script language=jscript src="'+BL_SURL+'/bl_start.js"></script>');
}
```

bl_style.css

```
.boxtext1 {color:#000197;font-family: Arial, Helvetica, sans-serif; font-size: 12;  
font-weight: bold;}  
.boxtext2 {color:#FF6600;font-family: Verdana, Arial, Helvetica, sans-serif; font-  
size: 16; font-weight: bold;}  
.boxtext3 {color:#000000;font-family: Verdana, Arial, Helvetica, sans-serif; font-  
size: 13; font-weight: bold;}  
.boxtext4 {color:blue}  
.boxtext5 {color:#000197;font-family: Arial, Helvetica, sans-serif; font-size: 11;}  
10 .boxtext6 {color:black;font-family: Arial, Helvetica, sans-serif; font-size: 12;}  
.boxtext7 {color:black;font-family: Arial, Helvetica, sans-serif; font-size: 11;}  
.pnlg {background-color:#fcff00;BACKGROUND-IMAGE:  
url(/buylites/bl/images/cornerg1.gif);BACKGROUND-POSITION:left  
top;BACKGROUND-REPEAT:no-repeat;cursor:hand}  
15 .pncg {background-color:#fcff00;cursor:hand}  
.pnrg {text-decoration:none;background-color:#fcff00;BACKGROUND-IMAGE:  
url(/buylites/bl/images/cornerg2.gif);BACKGROUND-POSITION:right  
bottom;BACKGROUND-REPEAT:no-repeat;cursor:hand}  
.pnlg {background-color:#fcff00;BACKGROUND-IMAGE:  
20 url(/buylites/bl/images/cornerlg1.gif);BACKGROUND-POSITION:left  
top;BACKGROUND-REPEAT:no-repeat;cursor:hand}  
.pnclg {background-color:#fcff00;cursor:hand}  
.pnrlg {text-decoration:none;background-color:#fcff00;BACKGROUND-IMAGE:  
url(/buylites/bl/images/cornerlg2.gif);BACKGROUND-POSITION:right  
bottom;BACKGROUND-REPEAT:no-repeat;cursor:hand}  
25 .pnl1 {color:#000000;background-color:#fcff00;BACKGROUND-IMAGE:  
url(/buylites/bl/images/corner1.gif);BACKGROUND-POSITION:left  
top;BACKGROUND-REPEAT:no-repeat;cursor:hand}  
.pnc1 {color:#000000;background-color:#fcff00;cursor:hand}  
30 .pnr1 {color:#000000;background-color:#fcff00;BACKGROUND-IMAGE:  
url(/buylites/bl/images/corner2.gif);BACKGROUND-POSITION:right  
bottom;BACKGROUND-REPEAT:no-repeat;cursor:hand}  
.l2 {background-color:#11ffff;BACKGROUND-IMAGE:  
url(/buylites/bl/images/corner1.gif);BACKGROUND-POSITION:left  
35 top;BACKGROUND-REPEAT:no-repeat;}  
.c2 {background-image: url(/buylites/bl/images/pattern1blue.gif);background-  
position: left bottom;background-repeat:repeat-x}  
.r2 {BACKGROUND-IMAGE:  
url(/buylites/bl/images/pattern1_dotblue.gif);BACKGROUND-POSITION:center  
40 top;BACKGROUND-REPEAT:no-repeat;}
```

```
.l3 {background-color:#11fff1;BACKGROUND-IMAGE:  
url(/buylites/bl/images/corner1.gif);BACKGROUND-POSITION:left  
top;BACKGROUND-REPEAT:no-repeat;cursor:hand}  
.c3 {background-color:#11fff1;cursor:hand}  
5 .r3 {text-decoration:none;background-color:#11fff1;BACKGROUND-IMAGE:  
url(/buylites/bl/images/corner2.gif);BACKGROUND-POSITION:right  
bottom;BACKGROUND-REPEAT:no-repeat;cursor:hand}  
.pns span {text-decoration:none;color:#000099;background-  
color="#CCCCCC;BACKGROUND-IMAGE:  
10 url(http://images.bizrate.com/site/un_b_bold.gif);BACKGROUND-POSITION:left  
bottom;BACKGROUND-REPEAT:repeat-x;}  
.pns span2 {background-color:#11fff1; cursor:hand}  
.pns span3 {background-color:#11fff1;}  
.pnul {text-decoration:none; color:#000000; background-color:#fcff00;  
15 BACKGROUND-IMAGE:  
url(/buylites/bl/images/underline_buylites.gif);BACKGROUND-POSITION:left  
bottom; BACKGROUND-REPEAT:repeat-x; cursor:hand;}
```

bl_vb.js

```
20 Function setInnnerHTML(elem, val)  
    ON ERROR RESUME NEXT  
    elem.innerHTML = val  
    setInnnerHTML = Err.Number  
END Function  
25 Function setOuterHTML(elem, val)  
    ON ERROR RESUME NEXT  
    elem.outerHTML = val  
    setOuterHTML = Err.Number  
30 END Function
```

bl_start.js

```
35 var blTry = 0;  
  
function blShowBLs()  
{  
    if (typeof(blHighlight) == 'function' && typeof(makeChanges) == 'function')  
40    {  
        blHighlight();
```

```
    }
    else
    {
        biTry++;
5       if (biTry < 10)
            setTimeout('biShowBLs()', BL_DELAY_RETRY);
    }
}

10 window.onload = biShowBLs;
```

© BizRate.com

Appendix C: Example of Client-Side Annotation and Recognition Code

Defiler.cpp

GnomeIEPageBroker.cpp

5 HTMLAnnotatorProxy.cpp

HTMLParser.cpp

IEDocUtils.cpp

ProperNounDB.cpp

10 Defiler.cpp

```
#include <stdafx.h>
#include "Defiler.h"
#include "HTMLParser.h"
#include "ProperNounDB.h"
15    #include "bizratestring.h"
#include <sstream>
#include "ebRegistryMgr.h"
#include "eBBarRegistryMgr.h"
#include "eBconst.h"
20    #include "eBScriptUtil.h"
#include "IEDocUtils.h"
```

using namespace std;

25 const int Defiler::AFFILIATE_ID = -2;

```
Defiler::Defiler(ProperNounTable *pnTable,const string &serverBaseURL) {
    this->pnTable=pnTable;           // This object takes ownership of the
pnTable.
```

30

```
eBRegistryMgr regMgrLM;
regMgrLM.InitAppKey( HKEY_LOCAL_MACHINE, eb_ebookle );
```

```
35    // Get Version servlet
CComBSTR redirectServlet;
regMgrLM.GetProfileString(RS_PRPOERNOUN,      RS_PN_REDIRECT_SERVLET,
&redirectServlet );
```

```
if (redirectServlet.Length() == 0)
{
    WriteLog(EBLOG_OBJECT, 0, "Defiler::Defiler: Invalid redirect servlet\n");
    return;
5    }
    _bstr_t bstrRedirectServlet = redirectServlet.m_str;
    annotationServletURL=serverBaseURL+((char *)bstrRedirectServlet);
}

10   Defiler::~Defiler() {
        delete pnTable;
    }

15   string *Defiler::AnnotateHTML(const string *originalHTML, IHTMLElement *elem,
CComPtr<IWebBrowser2> pIWebBrowser, IDispatch* pDisp, bool &bAddSpanTag) {
        string *annotatedHTML = NULL;
        AnnotationResult retval=UNCHANGED;
20        HTMLParser parser(*originalHTML);
        const DocumentWordList *docWords=parser.GetDocumentWords();

        ANNOTATION_SET *annSet=GetAllAnnotations(docWords,originalHTML);
        HRESULT hinsertedCode;
25        if(annSet->size()>0)
        {
            if (!bAddSpanTag)
            {
                CComPtr<IWebBrowser2> pMainWebBrowser;
30                pDisp->QueryInterface(IID_IWebBrowser2,(void **)&pMainWebBrowser);
                    hinsertedCode = InsertProperNounScript(pIWebBrowser,
pMainWebBrowser);
                    if(hinsertedCode == S_OK)
                        bAddSpanTag = true;
35            }
            if(bAddSpanTag)
                annotatedHTML=InsertPopup(annSet,originalHTML, elem,
pIWebBrowser, pDisp);
            }
40        }

        delete annSet;
        return annotatedHTML;
    }

45   HRESULT Defiler::InsertProperNounScript(CComPtr<IWebBrowser2> pIWebBrowser,
IDispatch* pDisp)
```

```
{  
    CComPtr<IWebBrowser2> pmainWebBrowser;  
    if(pDisp)  
        pDisp->QueryInterface(IID_IWebBrowser2,(void **)&pmainWebBrowser);  
    5      CComPtr<IHTMLDocument2> pdoc;  
    if(pmainWebBrowser)  
    {  
        CComPtr<IDispatch> pdispDoc;  
        pmainWebBrowser->get_Document(&pdispDoc);  
        10     pdispDoc->QueryInterface(IID_IHTMLDocument2,(void **)&pdoc);  
    }  
    if(!pdoc)  
        return S_FALSE;  
    CComVariant variantList[1];  
    15     variantList[0].vt = VT_DISPATCH;  
    pdoc->QueryInterface(IID_IDispatch,(void *)(&(variantList[0].pdspVal)));  
    CComVariant vResult;  
    eBScript::ExecFunc( pIWebBrowser, L"popupx", 1, variantList, &vResult );  
    if (vResult.vt == VT_I4 && vResult.intVal == 1)  
    20     return S_OK;  
    else  
        return S_FALSE;  
  
}  
25  
string *Defiler::InsertPopup(ANNOTATION_SET *annSet,const string *originalHTML,  
                             IHTMLElement *elem, CComPtr<IWebBrowser2> pIWebBrowser,  
                             IDispatch* pDisp) {  
    30     ostringstream oss;  
  
    ANNOTATION_SET::const_iterator iter=annSet->begin();  
    ANNOTATION_SET::const_iterator theEnd=annSet->end();  
  
    35     int nextPlaceToCopy=0;  
  
    while(iter != theEnd) {  
        const AnnotationWrapper &wrapper=*iter;  
        const Annotation *annotation=wrapper.GetAnnotation();  
        int startPos=annotation->GetStartIndex();  
        40     int endPos=annotation->GetEndIndex();  
  
        string htmlIPN = originalHTML->substr(startPos,endPos-startPos);  
  
        45     string htmlStart = originalHTML->substr(nextPlaceToCopy,startPos-  
nextPlaceToCopy);  
  
        oss << htmlStart;
```

```
        string *annotationURL=GenerateAnnotationURL(*annotation,htmlPN);

        CComBSTR bstrSpan;
        GetSpanHTML(piWebBrowser, pDisp, htmlPN, *annotationURL, bstrSpan);
5
        if (bstrSpan.Length() > 0)
        {
            _bstr_t tmpSpan = bstrSpan.m_str;
            oss << (char *)tmpSpan;
10
        }

        //string outputString = string(oss.str());
        //WriteLog(EBLOG_OBJECT, 0, "Defiler::InsertPopup: All_HTML=%s\n",
        outputString.c_str());
15
        nextPlaceToCopy=endPos;
        iter++;
    }

20    oss << originalHTML->substr(nextPlaceToCopy);

        //string outputString = string(oss.str());
        //WriteLog(EBLOG_OBJECT, 0, "Defiler::InsertPopup: All_HTML=%s\n",
        outputString.c_str());
25
        string annotatedHTML = oss.str();
        IEDocUtils::SetHTML(elem, annotatedHTML.c_str());

        return new string(oss.str());
30 }

HRESULT Defiler::GetSpanHTML(CComPtr<IWebBrowser2> piWebBrowser, IDispatch*
35                                const string &strPN, const string &urlPN, CComBSTR
                                &bstrSpan)

{
    int retval=0;

40    CComVariant variantList[3];

    CComBSTR bstrURL = urlPN.c_str();
    variantList[0].vt = VT_BSTR;
    variantList[0].bstrVal = bstrURL.Copy();
45
    CComBSTR bstrPN = strPN.c_str();
    variantList[1].vt = VT_BSTR;
```

```
variantList[1].bstrVal = bstrPN.Copy();

variantList[2].vt = VT_DISPATCH;
pDisp->QueryInterface(IID_IDispatch,(void **)&(variantList[2].pdispVal));
5
    CComVariant vResult ;
eBScript::ExecFunc( pIWebBrowser, L"getSpanTag", 3, variantList, &vResult );

10     if (vResult.vt == VT_BSTR)
{
    bstrSpan = vResult.bstrVal;
    return S_OK;
}
15     else
        return S_FALSE;
}

20 ANNOTATION_SET      *Defiler:: GetAllAnnotations(const      DocumentWordList
*docWords,const string *originalHTML) {
    ANNOTATION_SET *set=new ANNOTATION_SET;

25     unsigned int numWords=docWords->NumWords();
     unsigned int currentWord=0;

30     while(currentWord<numWords) {
        ProperNounTable *curTable=pnTable;
        const DocumentWord *firstWord=docWords->GetWord(currentWord);
        const DocumentWord *lastWord=firstWord;

35     char firstChar=originalHTML->at(firstWord->GetStringIndex());
     if( (::isUpperCaseLetter(firstChar)) || (::isNumber(firstChar)) ) {
            // Only check for annotations if the first character is
            // upper case or a number.
        while(true) {
            if(currentWord>=numWords)
                break;
40             const      DocumentWord      *tmpWord=docWords-
>GetWord(currentWord);
            const string *word=tmpWord->GetWord();

45             ProperNounTable      *nextTable=curTable-
>GetTableForWord(*word,false);
            if(nextTable==NULL)
                break;
```

```
        curTable=nextTable;
        lastWord=tmpWord;
        currentWord++;
5      }
    }

    if(curTable==pnTable) {
        // Never annotate at the root table. Skip over this word
10       // becayse it matched nothing.
        currentWord++;
    } else {
        // Okay, so some words matched. See if there is a proper noun
        // associated with the last table that contained a proper noun
15       // word.
        const ProperNoun *properNoun=curTable->GetProperNoun();
        if(properNoun!=NULL) {
            // Found a proper noun match. Add the annotation.
            Annotation annotation;
            annotation.SetAssociatedProperNoun(properNoun);
            annotation.SetStartIndex(firstWord->GetStringIndex());
            annotation.SetEndIndex(lastWord-
>GetStringIndex()+lastWord->GetWord()->size());
            set->insert(AnnotationWrapper(&annotation));
20
25       }
    }

    return set;
30 }

35 string *Defiler::GenerateAnnotationURL(const Annotation &annotation,string Keywords )
{
    const ProperNoun *pn=annotation.GetAssociatedProperNoun();
    ostringstream oss;

40    oss << annotationServletURL << "?keyword=" << Keywords << "&aff_id=" <<
AFFILIATE_ID
        << "&noun_id=" << pn->GetId() << "&fb=1&rf=bar" << pn->GetId() << "-
aff_id=" ;

45    return new string(oss.str());
}
```

GnomeIEPageBroker.cpp

```
#include <stdafx.h>
#include "GnomeIEPageBroker.h"
5 #include "IEDocUtils.h"
#include "GnomeConfig.h"
#include "ListLoader.h"
#include <string>
#include "eBScriptUtil.h"
10 using namespace std;

15 //Defiler *GnomeIEPageBroker::defiler = NULL;
//int GnomeIEPageBroker::defilerRefCount = 0;

20 //BRLock *GnomeIEPageBroker::refCountLock = LockFactory::MakeLock();

25 GnomeIEPageBroker::GnomeIEPageBroker(GnomeBaseConfig      *config,      int
pnListServerVersion) {
    // This object takes ownership of config.
    defiler=NULL;
    this->config=config;
    Init(pnListServerVersion);

30 }

35 GnomeIEPageBroker::~GnomeIEPageBroker() {
    delete annotatorProxy;
    delete config;

    // Decrease the reference count on the defiler. If the count is
    // zero, delete it; otherwise, keep it around because other people
    // are using it.
40    /*refCountLock->Lock();

        defilerRefCount--;
        if(defilerRefCount==0) {
            */
45        if(defiler)
            delete defiler;
        defiler=NULL;
```

```
    ////}

    ///refCountLock->Unlock();
}

5

int GnomeIEPageBroker::ProcessAllFrames(CComPtr<IWebBrowser2> pIWebBrowser,
IDispatch * pDisp) {
    if( (defiler == NULL) || !pDisp )
10    {
        WriteLog("GnomeIEPageBroker::ProcessAllFrames: annotations off\n");
        return 0;                                // Disabled annotations.
    }
    CComPtr<IWebBrowser2> pmainWebBrowser;
15    if(pDisp)
        pDisp->QueryInterface(IID_IWebBrowser2,(void **) &pmainWebBrowser);
    CComPtr<IHTMLDocument2> pdoc;
    if(pmainWebBrowser)
        {
20        CComPtr<IDispatch> pdispDoc;
        pmainWebBrowser->get_Document(&pdispDoc);
        pdispDoc->QueryInterface(IID_IHTMLDocument2,(void **) &pdoc);
    }
    if(!pdoc)
25        return 0;
    CComVariant variantList[1];
    variantList[0].vt = VT_DISPATCH;
    pdoc->QueryInterface(IID_IDispatch,(void **) &(variantList[0].pdispVal));
    CComVariant vResult;
30

    eBScript::ExecFunc( pIWebBrowser, L"isValidtoAnnotate", 1, variantList,
&vResult );
    if (vResult.vt == VT_I4 && vResult.intVal == 1)
        {
35        ProcessPage(pdoc,true, pDisp, pIWebBrowser);
    }
    return 0;
}
40 int GnomeIEPageBroker::ProcessPage(IHTMLDocument2 *doc,bool updateURLList,
IDispatch* pDisp, CComPtr<IWebBrowser2> pIWebBrowser) {
    // Annotate a single page.
    WriteLog("GnomeIEPageBroker::ProcessPage: in\n");
    /*if(updateURLList) {
        char *url=IEDocUtils::GetURL(doc);
45        urlList.AddURL(url);
        delete[] url;
    }*/
}

45
```

```
// Annotate the document.  
IEDocUtils::WalkThroughDocument(annotatorProxy, doc, pIWebBrowser, pDisp);  
  
5     return 0;  
}  
  
10    //bool GnomeIEPageBroker::Enabled() {  
11        //    return config->GetAnnotationsEnabled();  
12    //}  
  
15    void GnomeIEPageBroker::Init(int pnListServerVersion) {  
16        annotatorProxy=NULL;  
  
17        if(defiler==NULL) {  
18            // Only do these steps if the defiler has not been set up yet.  
19            string *serverURLBase=config->GetServerURLBase();  
20            ListLoader loader(*serverURLBase);  
21            delete serverURLBase;  
22            int clientPNListVer = config->GetPNListVersion();  
23            if(pnListServerVersion > clientPNListVer) {  
24                // A newer version is available.  
25                // Load the proper noun list from the network.  
26                loader.GetProperNounsFromURL(new  
27                    AsynchronousLoadCallback(this),pnListServerVersion);  
28            } else {  
29                // A newer version is not available. Load it from the  
30                // filesystem.  
31                string *cacheFilename=config->GetCacheFilename();  
32                ProperNounTable  
33                *pnTable=loader.GetProperNounsFromFile(*cacheFilename);  
34                delete cacheFilename;  
35                if(pnTable==NULL) {  
36                    // Exception.  
37                    // Something happened to the proper noun list in the  
38                    // filesystem. Load it from the network.  
39                    loader.GetProperNounsFromURL(new  
40                        AsynchronousLoadCallback(this),pnListServerVersion);  
41                } else {  
42                    SetDefiler(pnTable);  
43                }  
44            }  
45        } else {  
46            SetAnnotatorProxy();  
47        }  
}
```

```
}

5 void GnomeIEPageBroker::SetAnnotatorProxy() {
    delete annotatorProxy;
    annotatorProxy=new HTMLAnnotatorProxy(defiler,config);

    ///refCountLock->Lock();
10   ///defilerRefCount++;
    ///refCountLock->Unlock();
}

15 void GnomeIEPageBroker::SetDefiler(ProperNounTable *pnTable) {
    // This object takes ownership of pnTable.
    delete defiler;
    string *annotationURLBase=config->GetAnnotationURLBase();
    defiler=new Defiler(pnTable,*annotationURLBase);
20   delete annotationURLBase;
    ///defilerRefCount=0;

    SetAnnotatorProxy();
}

25 int GnomeIEPageBroker::TakeLoad(ProperNounTable *pnTable,const std::string
    *pnText,unsigned int latestVersion) {
    // Create a new defiler from the proper noun table. Then, store the
30   // latest proper noun list version and the list text in a file.

    // set the latest version in the registry
    eBRegistryMgr regMgrLM;
    regMgrLM.InitAppKey( HKEY_LOCAL_MACHINE, eb_eboodle );
    wchar_t wc[100];
    35   _itow( latestVersion , wc, 10);
    regMgrLM.WriteString(RS_PRPOERNOUN,
    RS_PN_CLIENT_PN_VERSION, wc);
    SetDefiler(pnTable);

40   string *cacheFilename=config->GetCacheFilename();
    ListLoader::StoreProperNounsInFile(*cacheFilename,*pnText);
    delete cacheFilename;

45   return 0;
}
```

HTMLAnnotatorProxy.cpp

```
#include <stdafx.h>
#include "HTMLAnnotatorProxy.h"
5 #include "IEDocUtils.h"
#include <string>
#include <sstream>
#include "eBScriptUtil.h"

10 using namespace std;

HTMLAnnotatorProxy::HTMLAnnotatorProxy(Defiler *defiler,GnomeBaseConfig *config)
{
15     this->defiler=defiler;
//RelevantTags::init();

    if(config->InDebugMode()) {
        string *cacheDir=config->GetCacheDirectory();
        debugHTMLWriter=new DebugHTMLWriter(cacheDir->c_str());
        delete cacheDir;
20    } else
        debugHTMLWriter=NULL;

25     annotationNum=0;
}

HTMLAnnotatorProxy::~HTMLAnnotatorProxy() {
30     delete debugHTMLWriter;
}

35 int HTMLAnnotatorProxy::Process(CComPtr<IWebBrowser2> pIWebBrowser, IDispatch*
pDisp, IHTMLElement *elem, bool &bAddSpanTag)
{
    USES_CONVERSION;

40    //CComBSTR tagNameBSTR;
    //HRESULT result=elem->get_tagName(&tagNameBSTR);

    //if(FAILED(result))
45    //    return -1;
```

```
//_bstr_t tagNameWrapper(tagNameBSTR.m_str); // tagNameWrapper takes
ownership of tagNameBSTR.

5    //if(!RelevantTags::IsRelevantTag(&tagNameWrapper))
//      return 0;

int elemCountAllowed = isElementValidity(pIWebBrowser,elem);
if( !elemCountAllowed )
    return 0;
10   //char *origHTML=IEDocUtils::GetHTML(elem); commented by RAJ
CComBSTR htmlBSTR;
elem->get_innerHTML(&htmlBSTR);

15   if( elemCountAllowed > 0 && elemCountAllowed != 1 )
{
    if(htmlBSTR.m_str && (wcslen(htmlBSTR.m_str) > elemCountAllowed))
        return 0;
20   }

if( (htmlBSTR.m_str != NULL) ){
//if( (htmlBSTR.m_str != NULL) && (wcslen(htmlBSTR.m_str)<4000) ){
    string *annotatedHTML=defiler-
25   >AnnotateHTML(&string(W2A(htmlBSTR.m_str)), elem, pIWebBrowser, pDisp,
bAddSpanTag);

    if(annotatedHTML != NULL) {
        // There is an annotation, so do it.
        if(debugHTMLWriter != NULL) {
            // Write out the changes to a file.
            string *filePrefix=GetHTMLWriterFilename();
            const char *filePrefixCStr=filePrefix->c_str();
            //debugHTMLWriter-
35   >WriteHTML(filePrefixCStr,origHTML,ORIGINAL_HTML);
            debugHTMLWriter-
>WriteHTML(filePrefixCStr,annotatedHTML->c_str(),ANNOTATED_HTML);
            delete filePrefix;
        }
40   //      WriteLog("HTMLAnnotatorProxy::Process: Before IEDocUtils::SetHTML\n");
//      SetSpanHTML(pIWebBrowser, pDisp, elem,annotatedHTML->c_str());
//      IEDocUtils::SetHTML(elem,annotatedHTML->c_str());
        delete annotatedHTML;
45   }
}
}
```

```
//delete[] origHTML;

    return 0;
}

5     int          HTMLAnnotatorProxy::isElementValidity(CComPtr<IWebBrowser2>&
pIWebBrowser,IHTMLElement * elem)
{
    if(!elem)
10        return 0;

        CComVariant variantRet;
        CComVariant variantList[1];
        variantList[0].vt = VT_DISPATCH;
15        elem->QueryInterface(IID_IDispatch,(void **)&(variantList[0].pdispVal));
        eBScript::ExecFunc( pIWebBrowser, L"isValidElement", 1, variantList, &variantRet );
        if(variantRet.vt == VT_I4 && variantRet.lVal == 1 )
            return 1;
        else if(variantRet.vt == VT_I4 && variantRet.lVal > 1 )
20            return variantRet.lVal ;
        else
            return 0;
    }

25     int HTMLAnnotatorProxy::SetSpanHTML(CComPtr<IWebBrowser2> pIWebBrowser,
                                         IDispatch*      pDisp,
IHTMLElement *elem,const char *html) {
    int retval=0;
    CComVariant variantList[3];
30
    CComBSTR strHTML = html;
    variantList[0].vt = VT_BSTR;
    variantList[0].bstrVal = strHTML.Copy();

35     variantList[1].vt = VT_DISPATCH;
    elem->QueryInterface(IID_IDispatch,(void **)&(variantList[1].pdispVal));

        variantList[2].vt = VT_DISPATCH;
        pDisp->QueryInterface(IID_IDispatch,(void **)&(variantList[2].pdispVal));
40
        eBScript::ExecFunc( pIWebBrowser, L"addSpanTag", 3, variantList, NULL );
        return retval;
    }

45     void HTMLAnnotatorProxy::SetDefiler(Defiler *defiler) {
        this->defiler=defiler;
    }
```

```
string *HTMLAnnotatorProxy::GetHTMLWriterFilename() {
    ostringstream oss;
5     oss << "X_" << (annotationNum++);
    return new string(oss.str());
}

10

/*****
*          *
* RelevantTags implementation.          *
*          *
*****/
```

```
/*
int RelevantTags::init() {
20    // Initialize the static members of the class if they have not already
    // been initialized.
    if(!isInitialized) {
        isInitialized=true;
        AddRelevantTags();
        return 1;
    }
25
    return 0;
}
30

bool RelevantTags::IsRelevantTag(const _bstr_t *tagWrapper) {
    return (tagset.find(*tagWrapper) != tagset.end());
}
35

void RelevantTags::AddRelevantTags() {
    // Add the relevant tags to the set.
    tagset.insert(_bstr_t("TD"));
40    // tagset.insert(_bstr_t("FORM"));
    tagset.insert(_bstr_t("P"));
    tagset.insert(_bstr_t("LI"));
    tagset.insert(_bstr_t("UL"));
}
45

TAG_BSTR_SET RelevantTags::tagset;
```

GANZLAW, PC
PO Box 10105
Portland, Oregon 97296
Phone: (503) 228-3641
Docket No.: BIZ/01-0003
Express Mail No: EL627039307US, Deposited June 12, 2001

```
bool RelevantTags::isInitialized = false;
```

```
*/
```

GANZLAW, PC
PO Box 10105
Portland, Oregon 97296
Phone: (503) 228-3641
Docket No.: BIZ/01-0003
Express Mail No: EL627039307US, Deposited June 12, 2001

HTMLParser.cpp

```
#include <stdafx.h>
#include "HTMLParser.h"
5 #include "bizratestring.h"
#include "StringTokenizer.h"

using namespace std;

10 const char *HTMLParser::DISCARDED_HTML_DELIMITERS = "\n\t";
const char *HTMLParser::INCLUDED_HTML_DELIMITERS = "<>";

15 const string HTMLParser::EXCLUDED_BEGIN_TAGS[] = {
    "<textarea>","<a>","<form>","<option>","<object>","<embed>"};
```

20 const string HTMLParser::EXCLUDED_END_TAGS[] = {
 "</textarea>","","</form>","</option>","</object>","</embed>"};

```
25 HTMLParser::HTMLParser(const string &html) {
    Parse(html);
30 }
```

```
35 HTMLParser::~HTMLParser() {
}
```

```
40 const DocumentWordList *HTMLParser::GetDocumentWords() {
    return &docWords;
}
```

```
45 int HTMLParser::Parse(const string &theHTML) {
    StringTokenizer
    tokenizer(theHTML,DISCARDED_HTML_DELIMITERS,INCLUDED_HTML_DELIMITER
S,true);
```

GANZLAW, PC
PO Box 10105
Portland, Oregon 97296
Phone: (503) 228-3641
Docket No.: BIZ/01-0003
Express Mail No: EL627039307US, Deposited June 12, 2001

```
int numTokens=tokenizer.NumTokens();
bool skipUntilEndOfTag=false;
int wordIndex=-1;
5    int excludedTagIndex=-1;

for(int i=0;i<numTokens;i++) {
    wordIndex++;
    const string *token=tokenizer.GetToken(i);
10   // Check if we're skipping over an anchor.

    // See if we're skipping over a tag whose body should be
    // completely exluded from annotation.
15   if(excludedTagIndex != -1) {
        // We're skipping over an exluded tag and its body
        if(IsExcludedEndTag(token,excludedTagIndex))
            excludedTagIndex=-1;
        continue;
20   }

    if( (excludedTagIndex=IsExcludedBeginTag(token)) != -1)
        continue;           // Skip over this tag's body.

25   // Check if we're skipping over a tag.
    if(skipUntilEndOfTag) {
        if(token->at(0)=='>')
            skipUntilEndOfTag=false; // Don't worry about getting
rid of the "<" -- it'll be trimmed in the code below.
30       else
            continue;
    }

35   if(token->at(0)=='<') {
        skipUntilEndOfTag=true;
        continue;
    }

40   // Add the word to the doc.
    int leftpos,rightpos;
    GetTokenTrimIndices(token,&leftpos,&rightpos);

    if(leftpos>rightpos)
        continue;
45   string *copy;
    if( (rightpos<token->size()-1) || (leftpos>0) )
```

```
        copy=new string(token->substr(leftpos,rightpos+1));
else
    copy=new string(*token);

5      docWords.AddWord(*copy,tokenizer.GetTokenPosition(i)+leftpos);
    delete copy;
}

return 0;
10 }

void HTMLParser::GetTokenTrimIndices(const string *token,int *leftPosIndex,int
*rightPosIndex) {
15     // Return the indices into token at which non-alphanumeric characters
// at either end stop.
// Trim non-alphanumeric characters on the right side.
int rightpos=token->size()-1;
while(rightpos>=0) {
20         char ch=token->at(rightpos);
        if( (!::isLetter(ch)) && (!::isNumber(ch)) )
            rightpos--;
        else
            break;
25     }

// Trim non-alphanumeric characters on the left side.
int leftpos=0;
while(leftpos<=rightpos) {
30         char ch=token->at(leftpos);
        if( (!::isLetter(ch)) && (!::isNumber(ch)) )
            leftpos++;
        else
            break;
35     }

*leftPosIndex=leftpos;
*rightPosIndex=rightpos;
}
40

int HTMLParser::IsExcludedBeginTag(const string *token) {

    int rightpo=token->size()-1;
45    // checks character by character for a matching tag ;
```

```
for(int i=0;i<numExcludedTags;i++) {
    const string *val = &EXCLUDED_BEGIN_TAGS[i] ;
    int leftpo=0;
    if ( val->size() == token->size() ) {
        while(leftpo<=rightpo) {
            char ch=token->at(leftpo);
            char ch1=val->at(leftpo);

            if ( (ch1 == ch || abs(ch -ch1) == 32 ) )
                leftpo++;
            else
                break ;
        }
        if ( leftpo == rightpo + 1 )
            return i;
    }
}
return -1;
}

bool HTMLParser::IsExcludedEndTag(const string *token,int tagIndex) {
// checks character by character for a matching tag ;
int rightpo=token->size()-1;
const string *val = &EXCLUDED_END_TAGS[tagIndex] ;
int leftpo=0;
while(leftpo<=rightpo) {
    char ch=token->at(leftpo);
    char ch1=val->at(leftpo);

    if ( (ch1 == ch || abs(ch -ch1) == 32 ) )
        leftpo++;
    else
        break ;
}
if ( leftpo == rightpo + 1 )
    return true;

// if(*token==EXCLUDED_BEGIN_TAGS[i])
// return i;

return false;
```

```
//return tolowercopy(token)==tolowercopy(&EXCLUDED_END_TAGS[tagIndex]);  
}
```

GANZLAW, PC
PO Box 10105
Portland, Oregon 97296
Phone: (503) 228-3641
Docket No.: BIZ/01-0003
Express Mail No: EL627039307US, Deposited June 12, 2001

IEDocUtils.cpp

```
#include <stdafx.h>
#include "IEDocUtils.h"
5 #include <ExDisp.h>
#include "bstrutil.h"
#include "eBScriptUtil.h"
using namespace std;

10 // Function prototypes.
void AddDocumentAndDescendents(IHTMLDocument2 *doc, DOCUMENT_VECTOR
*vec);

15 IHTMLDocument2 *IEDocUtils::GetMainDocument(IWebBrowser2 *browser) {
    // Return the main document of the web browser.
    IHTMLDocument2 *doc=NULL;
    IDispatch *dispatch;
20
    HRESULT result= browser->get_Document(&dispatch);
    if(SUCCEEDED(result)) {
        result= dispatch->QueryInterface(IID_IHTMLDocument2,(void **) &doc);
        dispatch->Release();
25    }

    return doc;
}

30 char *IEDocUtils::GetURL(IHTMLDocument2 *doc) {
    char *theURL = NULL;

    BSTR urlBSTR;
35    HRESULT result= doc->get_URL(&urlBSTR);
    if(SUCCEEDED(result)) {
        theURL= getCharArrayFromBSTR(urlBSTR);
        ::SysFreeString(urlBSTR);
        }
40    return theURL;
}

45 char *IEDocUtils::GetHTML(IHTMLDocument2 *doc) {
    // Return the HTML of the document.
    char *html = NULL;
```

```
IHTMLElement *body;
HRESULT result=doc->get_body(&body);

if(SUCCEEDED(result)) {
    5      html=IEDocUtils::GetHTML(body);
            body->Release();
    }

    return html;
10 }
```

```
char *IEDocUtils::GetHTML(IHTMLElement *elem) {
    // Get the HTML of the specified element. This method returns NULL
15    // if there is an error. Destroy the character array that is returned
    // when you're finished with it.
    char *theHTML=NULL;
    BSTR htmlBSTR;
    HRESULT result=elem->get_innerHTML(&htmlBSTR);
20
    if( (SUCCEEDED(result)) && (htmlBSTR != NULL) ) {
        theHTML=getCharArrayFromBSTR(htmlBSTR);
        ::SysFreeString(htmlBSTR);
    }
25
    return theHTML;
}
```

```
30 int                 IEDocUtils::WalkThroughDocument(HTMLElementProcessor
*processor,IHTMLDocument2 *doc, CComPtr<IWebBrowser2> piWebBrowser,
IDispatch* pDisp) {
    // Go through every element in the document and run the specified
    // processor on it.
35 WriteLog("IEDocUtils::WalkThroughDocument: in\n");

    int retval=0;
    bool bAddSpanTag=false;
    IHTMLElementCollection *collection;
40    HRESULT result=doc->get_all(&collection);
    if(SUCCEEDED(result)) {
        long numItems;
        result=collection->get_length(&numItems);

45    for(long i=0;i<numItems;i++) {
        // Iterate through each element and send it to the processor.
        VARIANT nameVar;
```

```
nameVar.vt=VT_I4;
nameVar.lVal=i;

VARIANT emptyVar = { 0 };

5      IDispatch *dispatch;
result=collection->item(nameVar,emptyVar,&dispatch);

10     if( (SUCCEEDED(result)) && (dispatch != NULL) ) {
          IHTMLElement *elem = NULL;
          result=dispatch->QueryInterface(IID_IHTMLElement,(void
**) &elem);
          if(SUCCEEDED(result)) {
15            processor->Process(pIWebBrowser, pDisp, elem,
bAddSpanTag);
            elem->Release();
          }
        }

20     dispatch->Release();
      }

25     collection->Release();
if(bAddSpanTag)
{
  CComPtr<IWebBrowser2> pmainWebBrowser;
  if(pDisp)
    pDisp->QueryInterface(IID_IWebBrowser2,(void
30   **)&pmainWebBrowser);
    CComPtr<IHTMLDocument2> pdoc;
    if(pmainWebBrowser)
    {
      CComPtr<IDispatch> pdispDoc;
      pmainWebBrowser->get_Document(&pdispDoc);
      pdispDoc->QueryInterface(IID_IHTMLDocument2,(void
35   **)&pdoc);
    }
    if(pdoc)
    {
      CComVariant variantList[1];
      variantList[0].vt = VT_DISPATCH;
      pdoc->QueryInterface(IID_IDispatch,(void
40   **)&(variantList[0].pdspVal));
      CComVariant vResult;
      eBScript::ExecFunc( pIWebBrowser, L>ShowBuyLitesTab",
45   1, variantList, &vResult );
}

1, variantList, &vResult );
```

```
        }

5      } else
         retval=-1;

        return retval;
    }

10

int IEDocUtils::SetHTML(IHTMLElement *elem,const char *html) {
    int retval=0;

15 //     if( NULL != strstr(html,<OBJECT") || NULL != strstr(html,<IFRAME"))
//         return retval;

        BSTR htmlBSTR=makeBSTR(html);
        elem->put_innerHTML(htmlBSTR);
20      ::SysFreeString(htmlBSTR);

        return retval;
    }

25 DOCUMENT_VECTOR *IEDocUtils::GetFrames(IWebBrowser2 *browser) {
    // Returns a vector of smart points to IHTMLDocument2 objects, each of which
    // is a frame in the web browser.
    DOCUMENT_VECTOR *vec=new DOCUMENT_VECTOR;
30

    IHTMLDocument2 *doc=GetMainDocument(browser);
    if(doc != NULL) {
        AddDocumentAndDescendents(doc,vec);
        doc->Release();
    }

35      }

        return vec;
    }

40 DOCUMENT_VECTOR *IEDocUtils::GetAllDescendentFrames(IHTMLDocument2 *doc)
{
    // Returns a vector of smart points to IHTMLDocument2 objects, each of which
    // is a frame under the document doc.
45      DOCUMENT_VECTOR *vec=new DOCUMENT_VECTOR;
        AddDocumentAndDescendents(doc,vec);
```

```
    return vec;
}

5   static void AddDocumentAndDescendents(IHTMLDocument2
*doc,DOCUMENT_VECTOR *vec) {
    // The method's client should call doc->Release() when this method returns.
    IHTMLDocument2Ptr docSmartPtr(doc);           // AddRef automatically
called internally.
10  vec->push_back(docSmartPtr);

    IHTMLFramesCollection2 *framesCollection;
    HRESULT result=doc->get_frames(&framesCollection);
    if(SUCCEEDED(result)) {
15    long numFrames;
    framesCollection->get_length(&numFrames);

        for(long l=0;l<numFrames;l++) {
            VARIANT indexVariant;
            indexVariant.lVal=l;
            indexVariant.vt=VT_I4;

            VARIANT frameVariant;

25        result=framesCollection->item(&indexVariant,&frameVariant);
        if(SUCCEEDED(result)) {
            IDispatch *dispatch=frameVariant.pdispVal;
            IHTMLWindow2 *frameWindow;
            result=dispatch->QueryInterface(IID_IHTMLWindow2,(void
30    **) &frameWindow);

            if(SUCCEEDED(result)) {
                IHTMLDocument2 *frameDoc;
                result=frameWindow->get_document(&frameDoc);

35            if(SUCCEEDED(result)) {
                AddDocumentAndDescendents(frameDoc,vec); // Add this child and its
descendents.
40            frameDoc->Release();
        }

45            frameWindow->Release();
        }

        dispatch->Release();
    }
}
```

```
    }  
    framesCollection->Release();  
}  
5 }
```

GANZLAW, PC
PO Box 10105
Portland, Oregon 97296
Phone: (503) 228-3641
Docket No.: BIZ/01-0003
Express Mail No: EL627039307US, Deposited June 12, 2001

ProperNounDB.cpp

```
#include <stdafx.h>
#include "ProperNounDB.h"
5 #include "StringTokenizer.h"
#include <iostream>
#include <fstream>
#include <sstream>
#include <assert.h>
10 using namespace std;

15 ProperNounDB::ProperNounDB() {
}

20 ProperNounDB::~ProperNounDB() {
}

25 void ProperNounDB::AddProperNoun(const ProperNoun &properNoun) {
    // Store a copy in the database.
    properNouns.push_back(properNoun);
}

30 const ProperNoun *ProperNounDB::GetProperNounByPhrase(string &phrase) const {
    // Return the proper noun matching the specified phrase, or return NULL
    // if no such proper noun exists in the database.
    PROPERNOUN_VECTOR::const_iterator iter=properNouns.begin();
    PROPERNOUN_VECTOR::const_iterator theEnd=properNouns.end();

    const ProperNoun *pn = NULL;
    while( (iter != theEnd) && (pn==NULL) ) {
        const ProperNoun &thePn=*iter;
        string *str=thePn.GetProperNounPhrase();
        if(*str==phrase)
            pn=&thePn;

40        delete str;
        iter++;
    }
}
```

```
    return pn;
}

5   const ProperNoun *ProperNounDB::GetProperNounById(int pnId) const {
    // Return the proper noun matching the specified proper noun id, or
    // return NULL if no such proper noun exists in the database.
    PROPERNOUN_VECTOR::const_iterator iter=properNouns.begin();
    PROPERNOUN_VECTOR::const_iterator theEnd=properNouns.end();
10
    const ProperNoun *pn = NULL;
    while( (iter != theEnd) && (pn==NULL) ) {
        const ProperNoun &thePn=*iter;
        if(thePn.GetId()==pnId)
            pn=&thePn;
15
        iter++;
    }
20
    return pn;
}

25   const PROPERNOUN_VECTOR *ProperNounDB::GetAllProperNouns() const {
    return &properNouns;
}

30   int ProperNounDB::PersistToFile(const char *filename) const {
    // Write the database to a file. It can be loaded later.
    ofstream out(filename);
35
    char *encodedStr=CreateStringForDB();
    out << encodedStr;
    delete[] encodedStr;
40
    out.flush();
    out.close();
45
    return 0;
}

int ProperNounDB::LoadFromFile(const char *filename) {
    ifstream in(filename);
```

```
string completeDB;
string str;
while(getline(in,str)) {
    5      completeDB.append("\n");
    completeDB.append(str);
}
in.close();
10     return CreateFromString(completeDB.c_str());
}

15 int ProperNounDB::CreateFromString(const char *encodedStr) {
    string rawData(encodedStr);

    // First, tokenize into lines containing proper nouns. Each is on
    // its own line. Lines are separated by a single newline character.
20     StringTokenizer tokenizer(rawData,"\n","",false);
    int numProperNouns=tokenizer.NumTokens();
    for(int i=0;i<numProperNouns;i++) {
        const string *properNounLine=tokenizer.GetToken(i);
        StringTokenizer lineTokenizer(*properNounLine,"|","",false);
        25     int numColumns=lineTokenizer.NumTokens();

        assert(numColumns > 1);

        // First column is the proper noun phrase. Subsequent columns
        // are proper noun ids.
        30     const string *properNounPhrase=lineTokenizer.GetToken(0);
        const string *properNounId=lineTokenizer.GetToken(1);

        35     int pnId=atoi(properNounId->c_str());
        AddProperNoun(ProperNoun(*properNounPhrase,pnId));
    }

    return 0;
}
40

char *ProperNounDB::CreateStringForDB() const {
    ostrstream oss;
    PROPERNOUN_VECTOR::const_iterator iter=properNouns.begin();
    45     PROPERNOUN_VECTOR::const_iterator theEnd=properNouns.end();

    while(iter != theEnd) {
```

```
const ProperNoun &pn=*iter;

// Write the proper noun phrase.
string *pnPhrase=pn.GetProperNounPhrase();
5 oss << (*pnPhrase) << "|";
delete pnPhrase;

// Write the proper noun id.
oss << pn.GetId();
10 // Newline line delimiter.
oss << "\n";

iter++;
15 }

oss << ends;

return oss.str();
20 }
```

© BizRate.com

GANZLAW, PC
PO Box 10105
Portland, Oregon 97296
Phone: (503) 228-3641
Docket No.: BIZ/01-0003
Express Mail No: EL627039307US, Deposited June 12, 2001

United States Patent & Trademark Office
Office of Initial Patent Examination -- Scanning Division



Application deficiencies found during scanning:

Page(s) _____ of _____ were not present
for scanning. (Document title)

Page(s) _____ of _____ were not present
for scanning. (Document title)

Scanned copy is best available.

Drawings.